

## Initial Deployment Process

The web application is bundled with an nginx container.

I have created a Kubernetes Cluster using kops in AWS. It currently has one master node and two worker nodes. (t2 micro Ec2 Instances)





<input type="checkbox"/>	Name ▾	Instance ID ▲	Instance Type ▾	Availability Zone ▾	Ir
<input type="checkbox"/>	master-us-east-1b.masters.yeshwanth.k8s.local	i-03e1810fedc59cc60	t2.micro	us-east-1b	
<input type="checkbox"/>	nodes.yeshwanth.k8s.local	i-048c9e82597ba9603	t2.micro	us-east-1b	
<input type="checkbox"/>	jenkins	i-064abeec6d36a4ac6	t3.small	us-east-1a	
<input checked="" type="checkbox"/>	nodes.yeshwanth.k8s.local	i-09274f2caf5153042	t2.micro	us-east-1b	

Fig 1 : The Master and worker nodes of the Kubernetes Cluster

There are 2 yaml files

### **Website\_deployment.yaml**

Here, the pods are deployed which has the containers running our web application. This creates 3 replicas of our pods. So even if any pod dies of any reason, kubernetes automatically spins up another to maintain our count.

The strategy of this deployment is set to rolling update to ensure zero downtime deployments.

*Command* : `kubectl apply -f website_deployment.yaml`

NAME	READY	STATUS	RESTARTS	AGE
website-64b96cd994-lt8kd	1/1	Running	0	3h
website-64b96cd994-s5d8g	1/1	Running	0	3h
website-64b96cd994-wf7rr	1/1	Running	0	3h

Fig 2 : The three pods which are running in the kubernetes cluster

### **Service\_lb\_website.yaml**

Now that the pods have been created, this yaml file creates a service (Type: Load Balancer) . This gives the public URL of our application.

*Command* : `kubectl apply -f service_lb_website.yaml`

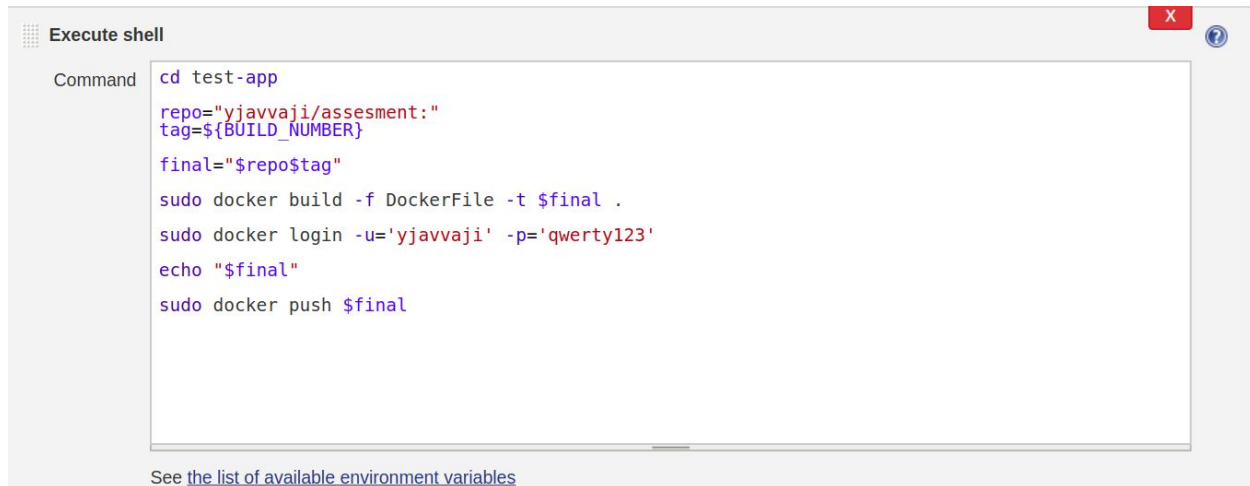
This Service is currently running at

[A387af5a82f2011e9b1e102368b90bba-1114145020.us-east-1.elb.amazonaws.com](https://A387af5a82f2011e9b1e102368b90bba-1114145020.us-east-1.elb.amazonaws.com)

### **Deployment process after code updates**

Suppose there is a new version of our web application and a new push happened to our git repository.

I have set up a jenkins build which automatically detects the change in the code and creates a new docker image and pushes to docker hub.

A screenshot of the Jenkins 'Execute shell' interface. The window has a title bar with a grid icon, the text 'Execute shell', a red close button with an 'X', and a help icon. Below the title bar, the word 'Command' is followed by a text area containing a shell script. The script sets environment variables for repository, tag, and final image name, then uses 'sudo' to build, login, and push a Docker image. At the bottom of the window, there is a link to 'the list of available environment variables'.

```
cd test-app

repo="yjavvaji/assesment:"
tag=${BUILD_NUMBER}
final="$repo$tag"

sudo docker build -f DockerFile -t $final .
sudo docker login -u='yjavvaji' -p='qwerty123'
echo "$final"
sudo docker push $final
```

See [the list of available environment variables](#)

Fig 3 : Jenkins shell script for pushing docker image

Now we just need to edit the deployment yaml file by renaming the image name with the new name and run the command given below.

*kubectl apply -f **website\_deployment\_new.yaml** --record*

## Jenkins

Jenkins username = user

Password = uuBBLKfjf1LR

Jenkins url = <http://ec2-3-87-156-73.compute-1.amazonaws.com/jenkins/>